TITLE:

Risa/Asir のifplot の改良と並列化の試み (Computer Algebra : Design of Algorithms, Implementations and Applications)

AUTHOR(S):

近藤, 祐史; 村尾, 裕一; 齋藤, 友克

CITATION:

近藤, 祐史 ...[et al]. Risa/Asir のifplot の改良と並列化の試み (Computer Algebra : Design of Algorithms, Implementations and Applications). 数理解析研究所講究録 2007, 1568: 185-191

ISSUE DATE:

2007-09

URL:

http://hdl.handle.net/2433/81208

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
第1568巻 2007年 185-191

Risa/Asirのifplotの改良と並列化の試み

近藤祐史　村尾裕一　齋藤友克
YUJI KONDOH　HIROKAZU MURAO　TOMOKATSU SAITO
詫間電波高専　電気通信大学　アルファオメガ

1 始めに

1.1 経緯と現状

Risa/Asirの零点表示機能ifplotは，1997年に提案されたアルゴリズムによって構成されている．このアルゴリズム[1, 2, 3]は，他の描画ソフトとは異なり数式処理システムの機能を前提に構成されている．そのため発表当時は，システムに対して比較的重い機能であった．しかし，表示される零点は誤差解析をする必要がなく常に正しい図形が描画されるため一定の評価はされてきた．

ifplotも開発当初から既に10年が経過し，当時と比べて計算機の処理能力は格段に向上してきた．現在においては，それほど重いプロセスというほどではなくなってきている．その一方，機能的に少々不足する点も指摘されてきている．そこで，新たな機能を付け加えifplotの改良を検討する．

検討する新たな機能としては，

- 不等式系の領域表示機能
- 3-D表示機能
- 並列処理の導入

である．

1.2 より適用範囲が広い指標関数の開発

ifplotは，描画を指標関数と呼ばれる描画空間Dを分割したCellと呼ばれる描画素から{0,1}への写像としてとらえている．描画をより高速かつ正しくあつかうためには，適切な指標関数の開発がifplotにおいて常に重要な問題である．

定義1 (Cellの定義)
C_k $(k = 1, \cdots, m)$がD上定義されたCellであるとは，$D = \bigcup_{k=1}^{m} C_k$，かつ$C_k^i \bigcap_{k \neq j} C_j^i = \phi$，各$C_k$のJordan測度はゼロではないとする．ここで$C_k^i$は，$Ce_k$の内点の全体とする．

定義2 (指標関数の定義)
D上定義されている関数fのCellの族$\{C_k\}$による指標関数χとは，$\chi : \{C_k\} \to \{0, 1\}$かつ$\chi(C_k) = 0$ならば$C_k$の任意の元$x$に対し$f(x) \neq 0$とする．

指標の定義よりifplotの描画は，零点が存在しない領域を消去する描画である．今回提案する指標関数は，これまでの指標関数とは異なり指標関数を組み合わせて判定する，複合指標関数と呼ぶべきものである．

2 不等式系の領域表示機能

2.1 不等式表示指標関数

ifplot は，零点の表示をする機能であるが，さまざまな方面から不等式系の解領域を表示する機能が必要であるとの意見があった．そこで ifplot の機能を一部修正し不等式系 $\{f_1 > 0, \cdots f_m > 0\}$ 解領域を表示する機能を付け加えた．この機能は，指標関数の定義を一部変更して

$$\chi(C_k) = 0 \quad \text{ならば} \quad f_i(C_k) < 0 \quad 1 \le i \le m$$

としたものである．実装等は単に ifplot のコードの一部を修正したことにより実現している．

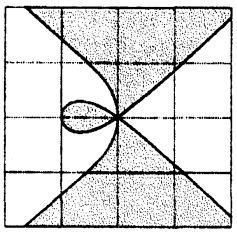
現状は以下のようになっている．

1. 現時点では，不等式系の領域描画は，unix 系でほぼ完成している．
2. インターフェイス等に関しては仮実装である．つまり変更する可能性がある．
3. windows 系ではまだ実装されていない．実装が X に強く依存した形式で構成されている関係である．

2.2 ineqn の実行例

$y^4 - x^4 + 2 * x * y^2 - x^3 > 0$ を表示するには，次のようにする．

```
ineqn(y^4-x^4+2*x*y^2-x^3,0x7cfc00,[x,-2,2],[y,-2,2],[600,600]);
```



ineqn 系列のコマンドは表示関数と領域操作関数とで出来ている．描画領域の and, or, xor, cp を引数とするいくつかのコマンドがある．また，領域を求めるアルゴリズムは ifplot の流儀と同様である．

3 3D–表示機能

3.1 現在における問題点

長年希望が多かった Risa/Asir に 3D–plot を導入することを考える．そのためには，3D–表示機能を実現するための問題点を明確化する必要がある．

指標関数の構成 3 変数以上の指標関数を構成することは，数学的に大変困難な問題である．

計算量の増大 ifplot の流儀に従えば理論上 ifplot の z 軸ピクセル分だけ計算量が増える．当然描画までの時間も新しい 3D 機能を考慮しなくても数百倍になる．

例えば，heart 関数は，現在のほとんどの計算機において 1 秒以下で表示できる．しかし，現在の ifplot の実装をそのまま敷衍して 3D 描画をおこなうと，数十秒かかる計算になる．これは，ユーザを待たせられる限界を越える．

現時点では，理論的な問題の解決とより一層の実装の効率化が必要である．

4 ifplotの高速化

3D描画のためだけでなく，より高次の多項式を描画する場合当然表示に必要とする時間は増大する．そのためifplotの高速化は常に重要な問題である．

ifplotを高速化するための解決策として次の2つが考えられる．

1. 並列処理の利用
2. 新しい指標関数の導入

本節では，前者について今後の予定を概説し，次節では後者について実験結果を交えて詳しく説明する．

4.1 グラフ表示における並列処理

現時点では，並列化は可能性と方法の検討等の段階である．

数式の零点表示とは，数式の数値化と数値データ列の2次元表示の問題に帰着できる．数式の数値化すなわち式の評価は，典型的なデータ並列処理の対象である．3Dであれば変数が3であるため最低でも3重ループが必要である．どのレベルのループを多重化できるかは，並列度をあげる上で重要である．

また，数式の計算においては，メモリの自動的な割当・開放は不可欠だが，並列処理環境下におけるメモリ管理は容易ではなく，その方法は自明ではない．本研究ではできるだけメモリ管理を行わないように，行うにしてもできるだけ簡単な方法をとるという方針で考えることにする．式の評価を単純に1回だけ行うのであればメモリの割当ては発生しないが，陰関数や変数が3個以上の式の描画では変数を順に減らしていくという方策をとるためメモリの管理が必要となる．一般には，並列処理環境下でのメモリ管理では，メモリ領域の確保要求やGCのスレッド間での競合の解消が必要である．

4.1.1 数式の数値化問題

上記のとおり，式の描画において $y = g(x)$ の数値化は割と自明だが，陰関数 $f(x, y) = 0$ の場合は問題が生ずる．例えば，式 $g_j(x) = f(x, y_j)$ や $h_i(y) = f(x_i, y)$ を作成するにはメモリ管理は不可欠である．しかし，描画の場合は生成される式は1変数に限られ，しかも，我々の応用では多項式と限定することができるので，1変数多項式を1次元配列で表現することにすれば一般的な意味でのメモリ管理を行う必要はなくなる．

4.1.2 多重ループと並列処理

式の評価は一般には多重ループとなる．即ち，例えば $f(x, y) = 0$ を次節の方法に従って描画する場合は，$x = x_1, ..., x_n$; $y = y_1, ..., y_m$; $f(x, y)$ の構成要素（多項式の項など）についての繰り返しの，少なくとも3重ループとなる．一般論として，オーバーヘッド(OpenMPの)を考慮すると，できるだけ外側のループを並列化したいのだが，上のメモリ管理と考えあわせて実現する必要がある．

基本的には，次のようにすればよいであろう．

1. 最外部の並列化 式 $g_j(x) = f(x, y_j)$ や $h_i(y) = f(x_i, y)$ のどちらかが多項式（or有理式．定型ならばよい）になる場合

- $g_j(x)$（または $h_i(y)$）を，予めスレッド毎に割り当てられた配列に作成する
- $x = x_1, ..., x_n$ に対する $g_j(x)$ の評価を，x の値の列を適宜ブロック化して行う（多点評価）

2. 外から二番目のループを並列化　式 $g_j(x) = f(x, y_j)$ と $h_i(y) = f(x_i, y)$ のどちらも前項の条件に当てはまらない場合

- $g_j(x)$ を求める
- $x = x_1, ..., x_n$ に対する $g_j(x)$ の評価を，x の値の列を適宜ブロック化し各ブロックをスレッドに割り当てて並列処理（それぞれでは多点評価）

多点評価については，式が多項式の場合は漸近的高速アルゴリズムの利用可能性と高速性についても検討する必要がある．

5　ifplotの新しいアルゴリズムの導入

5.1　解の非存在領域の発見

ifplot 流の描画とは，解の非存在領域を発見して描画領域から除いていく手法である．この判定を指標関数の計算と呼ばれている．

アルゴリズムを高速化するためには，より早く非存在領域を発見する必要がある．そのため，今回複合指標を導入した．複合指標とは，2 つ以上の指標関数を状況に応じて使い分ける指標関数である．今回は，区間数を利用する指標関数と符号判定の指標関数を組み合わせる複合指標関数としている．

5.1.1　提案する複合指標関数

区間数を利用する指標関数は，一定の領域に解が存在しないことを判定するためには比較的軽い方式である．そのため，まず始めに大きな領域で判定をおこない，解の存在する可能性があれば領域を細分し判定を進める方針とした．存在を否定できない領域が一定以下の大きさになった場合，符号判定指標関数による判定をおこなった．この符号判定指標関数は，ifplot において利用している指標関数である．

アルゴリズムの構成は次のようになっている．

アルゴリズムの基本

1. 領域を 4 つの矩形領域に分割する．
2. 個々の領域を左下を原点になるように平行移動する．
3. 個々の領域を 1 つの区間領域として区間演算する．
4. (a) 結果の区間が 0 を含まない
   $\Longrightarrow$ 当該領域を判定から除く．
   (b) 結果が 0 を含む
   $\Longrightarrow$ 1. から繰り返す．

5.2　動作速度等の判定

今回は，理論的な計算量の追求よりも，実際の実行速度をもとに動作の可否を決定した．テストデータとしては，Risa/Asir に含まれる ifplot の多項式 spade, heart, diamond, club を利用した．

5.3 実験の手順

毎回 Risa/Asir を立ち上げなおし以下のようなコマンドをスクリプトから投入した.

```
load("ifplot")$
ctrl("itvplotsize",65)$
itvplot(C)$
```

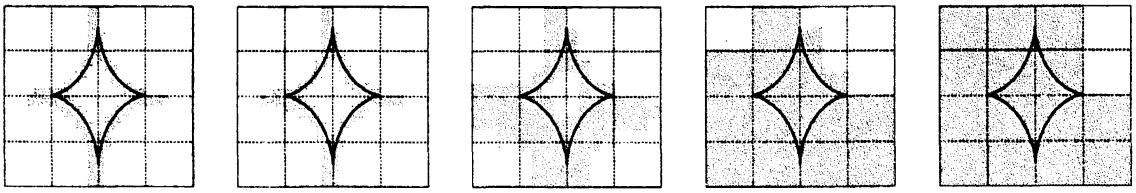
ここで 65 は，以下で示す itvplotsize と呼ばれるもので，区間指標関数の適用を 65dots 平方で打ち切り，符号判定指標関数に移行するよう指示するものである.

- `itvplot` を $2^n + 1$ とし $3 \leq n \leq 7$ の範囲を計測した.
- 実験は，各々10 回繰り返しその平均値を求め表示している.
- 実験結果は，Risa/Asir のソースに手を入れ cpu, gc の時間を別途表示させた.
- 特に区間指標関数で消去したか，本来の符号判定指標関数で消去したか区別できるように臨時に符号判定指標関数で消去した領域はグレイで表示するようにしてある.

5.4 実験の結果

5.4.1 diamond の実験結果

消去状況



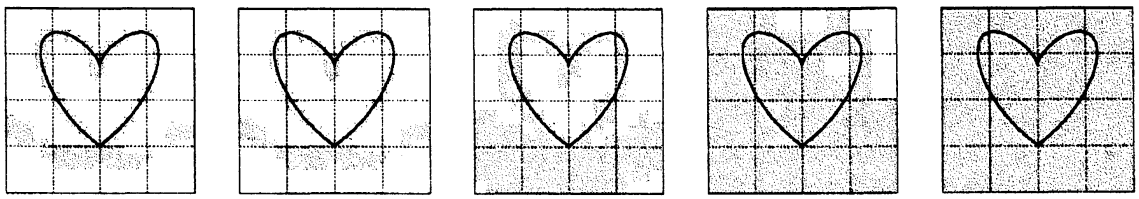
itvplotsize: $2^3 + 1$ itvplotsize: $2^4 + 1$ itvplotsize: $2^5 + 1$ itvplotsize: $2^6 + 1$ itvplotsize: $2^7 + 1$

実行時間

指標関数	itvplot					ifplot
移行数値	2^3+1	2^4+1	2^5+1	2^6+1	2^7+1	
cpu	0.134	0.131	0.119	0.137	0.138	0.133
gc	0.044	0.044	0.041	0.047	0.050	0.045
total	0.178	0.175	0.159	0.184	0.188	0.177

5.4.2 heart の実験結果

消去状況



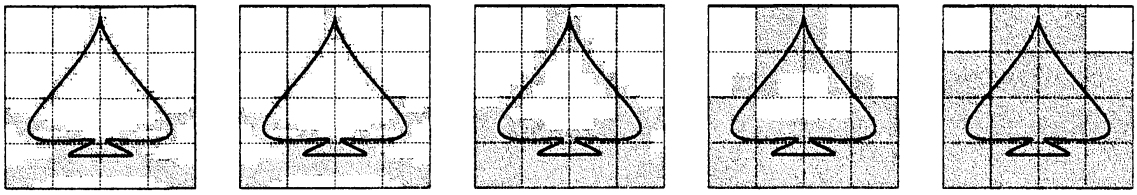
itvplotsize: $2^3 + 1$ itvplotsize: $2^4 + 1$ itvplotsize: $2^5 + 1$ itvplotsize: $2^6 + 1$ itvplotsize: $2^7 + 1$

実行時間

指標関数	itvplot					ifplot
移行数値	2^3+1	2^4+1	2^5+1	2^6+1	2^7+1	
cpu	0.170	0.169	0.148	0.160	0.157	0.222
gc	0.059	0.058	0.052	0.058	0.056	0.078
total	0.229	0.227	0.201	0.218	0.214	0.300

5.4.3 spade の実験結果

消去状況



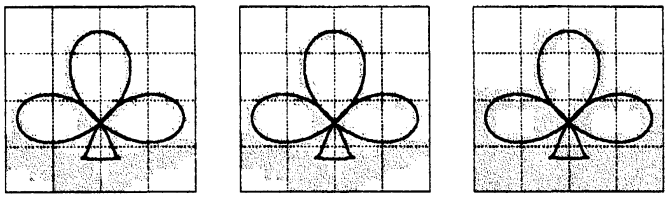
itvplotsize: 2^3+1 itvplotsize: 2^4+1 itvplotsize: 2^5+1 itvplotsize: 2^6+1 itvplotsize: 2^7+1

実行時間

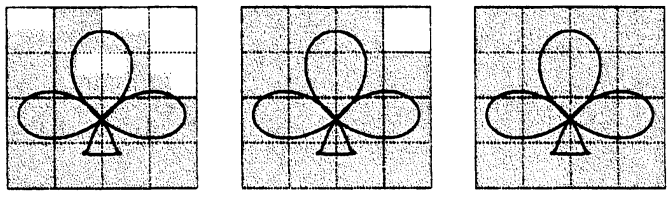
指標関数	itvplot					ifplot
移行数値	2^3+1	2^4+1	2^5+1	2^6+1	2^7+1	
cpu	0.299	0.295	0.196	0.170	0.185	0.294
gc	0.112	0.111	0.074	0.064	0.070	0.107
total	0.411	0.406	0.270	0.234	0.256	0.401

5.4.4 club の実験結果

消去状況



itvplotsize: 9 itvplotsize: 17 itvplotsize: 33



itvplotsize: 65 itvplotsize: 129 itvplotsize: 257

実行時間

指標関数	itvplot						ifplot
移行数値	2^3+1	2^4+1	2^5+1	2^6+1	2^7+1	2^8+1	
cpu	0.490	0.490	0.301	0.259	0.246	0.259	0.368
gc	0.208	0.209	0.121	0.104	0.099	0.104	0.135
total	0.699	0.699	0.422	0.362	0.344	0.364	0.503

6 結論と今後の展望

ifplot に区間数を取り入れた複合指標関数を取り入れた．実験結果としては，

suit	比率	itvplotsize
diamond	0.902	2^5+1
heart	0.668	2^5+1
spade	0.584	2^6+1
club	0.685	2^7+1

となり itvplotsize を適切に設定すれば良い結果が得られた．少なくとも itvplotsize が一定の数 $n < 33$ でなければ高速化されている．

経験的には，ほぼ半数の領域を消去した時であった．しかし，このことは，理論的には解明が進んでいない．並列化とあわせ今後の課題である．

参 考 文 献

[1] Saito, T. An extension of sturm's theorem to two dimensions. *Proceedings of the Japan Academy*, Vol. 73 A, pp. 18–19, 1997.

[2] 齋藤友克・近藤祐史・三好善彦・竹島卓. Displaying real solution of mathematical equations. 『数式処理』, Vol. 6, No. 2, pp. 2–21, 1998.

[3] Saito, T., Kondoh, Y., Miyoshi, Y., and Takeshima, T. Faithful plotting of real curves defined by bivariate rational polynomials. 情報処理学会論文誌, Vol. 41, No. 4, pp. 1009–1017, 2000.